Electronic Compass Test Plan

Cody Gabriel

Ben Huntsman

John Maseda

Michael Walton

Table of Contents

Introduction

This document describes how to test the electronic compass.

Test Setup:

# Overview of Test Cases

**Acceptance Test**

The purpose of this test is to validate and verify that our system meets the design requirements. For instance, for one of our design requirements is that our device displays the heading in degrees within 5° of actual. To test for this we will need to have another compass to compare the output to. For the acceptance test, we will be referencing the engineering requirements, not the marketing requirements.

|  |  |
| --- | --- |
| **Engineering Requirements** | **Justification** |
| 1. The system should be accurate to within 5° of true orientation. | To be a viable compass for use it should have an acceptable accuracy that would not result in an individual or robot being lost at long distances. |
| 2. The system should have a switch to power on the device and within a short time be ready for use. | To save battery life, the user should be able to turn the device on and off. The re-initialization of the device should require as little time as possible so that the user can get their heading and proceed with very little lost time. |
| 3. The LCD display will show a heading (i.e. "N", "SW") and degrees from North between 0 and 360 degrees. | Without an actual moving compass face, the user will need to easily understand the output of the device. Having eight segments it will be easy for the user to visualize their bearing. |
| 4. Should be prepared for later versioning in firmware that will give additional options and restore to factory default through a USB interface. | If an unforeseen software glitch were to arise the device will still possibly be useful if restored to factory settings. The device should be capable of a software update to correct any unforeseen glitches. |
| 5. LCD screen will have a back light so that the compass can be used in light or dark. | Needs to be used in varying conditions. |

1. The group will use our compass and compare the output with another physical compass app to ensure that our design will stay within a 5° tolerance as set by the project requirements
2. To test for this requirement, the group has implemented a switch into the design that disconnects power to the system. To test for the amount of time it takes to initialize the device, a stopwatch (on a smartphone) will be used. A finite initialization was not defined in the requirements.
3. This requirement is self-explanatory, the LCD displays the correct information or it does not. This can again be tested with the use of another reference compass to verify that the LCD is displaying the correct heading in degrees as well as the arrow pointing in the right direction.
4. To verify the 4th requirement, we can have two sets of coding. One will be missing some functionality and the other will include it. To verify the requirement, we will implement the change via USB.

The group may be getting rid of the 5th requirement. But if we don’t, we can simply toggle the LCD backlight in a dark room to confirm visibility under dark conditions.

**Integration Test**

The purpose of an integration test is to test the sub-modules, which have been individually tested previously, to see how they interact with each other. This is where we verify that the communication between modules is correct. For example, we should test the output of the PMIC under three conditions: battery only, USB only, and both. With the implemented test points we can measure the output of the PMIC with a multi-meter. We still need to consider the engineering requirements when developing the integration tests.

PMIC – The Power Management Integrated Circuit (PMIC) outputs a constant and stable 3.3V to the system. The group will need to verify that all modules receive power from the PMIC. This is different than testing the output of the PMIC as it is loaded while under operation. The group needs to ensure that all modules get power.

Microcontroller – Testing for this should be ongoing while writing the code. Ben or Michael will need to help me write this up.

Magnetometer/Accelerometer – The LSM303DLH communicates with the ATMega32U4 via the two-wire interface (TWI). Verification can be done by connecting a logic analyzer to the bus before powering on the device to observe the transactions between the ATMega32U4 master and LSM303DLH slave. The software writes to two separate configuration registers on the LSM303DLH, and then continuously reads from the LSM303DLH output registers. The transactions that should be observed are the following:

/\* Write to the accelerometer configuration register \*/

26 /\*

27 \* We wish to write one byte (0x27) to the accel config register

28 \*

29 \* The flow should be:

30 \* Master: START

31 \* Master: SAD + W

32 \* Slave: SAK

33 \* Master: SUB

34 \* Slave: SAK

35 \* Master: DATA

36 \* Slave: SAK

37 \* Master: STOP

38 \*/

/\* Write to the magnetometer configuration register \*/

102 /\*

103 \* We wish to write one byte (0x00) to the mag config register

104 \*

105 \* The flow should be:

106 \* Master: START

107 \* Master: SAD + W

108 \* Slave: SAK

109 \* Master: SUB

110 \* Slave: SAK

111 \* Master: DATA

112 \* Slave: SAK

113 \* Master: STOP

114 \*/

/\* Read 6 bytes from the accelerometer \*/

197 /\*

198 \* We need to read 6 bytes from the accelerometer.

199 \*

200 \* The flow for this is:

201 \* Master: START, SAD + W

202 \* Slave: SAK

203 \* Master: SUB

204 \* Slave: SAK

205 \* Master: RESTART, SAD + R

206 \* Slave: SAK

207 \*

208 \* The following iterates for 5 bytes in our case:

209 \* Slave: DATA

210 \* Master: MAK

211 \* ...

212 \*

213 \* Slave: DATA

214 \* MASTER: NMAK, STOP

215 \*/

/\* Read 6 bytes from the magnetometer \*/

346 /\*

347 \* We need to read 6 bytes from the magnetometer.

348 \*

349 \* The flow for this is:

350 \* Master: START, SAD + W

351 \* Slave: SAK

352 \* Master: SUB

353 \* Slave: SAK

354 \* Master: RESTART, SAD + R

355 \* Slave: SAK

356 \*

357 \* The following iterates for 5 bytes in our case:

358 \* Slave: DATA

359 \* Master: MAK

360 \* ...

361 \*

362 \* Slave: DATA

363 \* MASTER: NMAK, STOP

364 \*/

LCD – Testing for this should be ongoing while writing the code. Ben or Michael will need to help me write this up.

**Unit Test**

The purpose of a unit test is to individually test a sub-module for its desired behavior. This is an isolated test of the sub-module and should utilize ‘stubs’ to model the behavior of the other sub-modules the module-under-test interfaces with. For example, to do a unit test on the magnetometer, the group can use the microcontroller as a stub to output a blinking LED, where the number of LED flashes is 1/10 of the degree heading.

PMIC – Test the PMIC output voltage under three conditions: with battery, with USB, and with both the battery and USB. The charging functionality will also have to be tested. The group will have to drain a battery, check the voltage with a multi-meter, then charge it, and check the voltage after some amount of time charging, say two hours.

Microcontroller – To verify that the microcontroller is operating as intended we can observe one of the several unused GPIOs that will be set to output a high.

Magnetometer/Accelerometer – If communication is occurring and the device is configured properly then the DRDY test point on the breakout board will be asserted. If a high on DRDY is observed then the sensor is properly storing output information in it's registers.

**Debugging**

Any problems that may arise in the operation of the device will make use of a logic analyzer or a mixed signal oscilloscope for debugging. Transactions on both the TWI and the SPI bus will be observed using the logic analyzer, and microcontroller operation can be observed using the MSO.

Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:** Michael Walton | | | | | | |
| **Test Case Name:** | LSM303DLH Sensor Output Verification | **Test ID #:** | LSM#1 | | | |
| **Description:** | The device will output a high to the DRDY test point indicating it is properly configured and placing acceleration and magnetic field readings on it's output registers. | **Type:** | Black box | | | |
| **Tester information** | | | | | | |
|  | **Name of tester:** |  | **Date:** |  | | |
| **Hardware Version:** | 1.0 | **Time:** |  | | | |
| **Setup:** | SCL and SDA are connected to the TWI pins of the ATMega32U4 on Port D pin 0 and Port D pin 1 respectively. INT1 is tied to Port F pin 0 and INT2 is tied to Port F pin 1. SA is tied to ground. Vin and GND are tied to VCC and ground respectively. | | | | | |
| **Step** | **Action** | **Expected result** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | Turn on the compass using the power switch | Power LED should light up |  |  |  |  |
| 2 | Probe the DRDY test point of the LSM303DLH | DRDY should be held high at around 3 volts |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:** Michael Walton | | | | | | |
| **Test Case Name:** | Nokia5110 Display Test | **Test ID #:** | LCD#1 | | | |
| **Description:** | The display will indicate the heading via the UI design | **Type:** | Black box | | | |
| **Tester information** | | | | | | |
|  | **Name of tester:** |  | **Date:** |  | | |
| **Hardware Version:** | 1.0 | **Time:** |  | | | |
| **Setup:** | The LCD backlight pin is connected to a transistor circuit that's activated by Port B pin 6 of the ATMega32U4. The SCE and GND pins are tied to ground. D/C# is wired to Port B pin 4, and RST is tied to Port B pin 5. SCK and MOSI are tied to the SCK and MOSI pins on the ATMega32U4 which are Port B 1 and 2 respectively. | | | | | |
| **Step** | **Action** | **Expected result** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | Turn on the compass using the power switch | The LED should blink once for less than a second and go low indicating there’s no audio file in the chip |  |  |  |  |
| 2 | Look at the LCD | The user interface is displayed on the screen that serves to indicate the current magnetic heading |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:** Cody Gabriel | | | | | | |
| **Test Case Name:** | LTC4067 PMIC Test | **Test ID #:** | PMIC#1 | | | |
| **Description:** | The PMIC supplies the power to the system. The PMIC can have the battery, USB, or both as inputs. The outputs of the PMIC are a constant 3.3 Vdc and the charging of the battery. | **Type:** | Black box | | | |
| **Tester information** | | | | | | |
|  | **Name of tester:** |  | **Date:** |  | | |
| **Hardware Version:** | 1.0 | **Time:** |  | | | |
| **Setup:** | The outputs of the PMIC must be tested with the three input combinations. The unit must output a constant 3.3 Vdc when connected to the battery, USB, or both. The PMIC must also charge the battery when its voltage has dropped below the 2.8 V lower threshold and stop charginge once the battery has a voltage of 4.2 Vdc. This test is to be conducted using a multimeter that is available in the labs on campus. | | | | | |
| **Step** | **Action** | **Expected result** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | Connect the battery only. | The PMIC should output 3.3 Vdc |  |  |  |  |
| 2 | Disconnect the battery, plug in the USB cable attached to a DC source (computer) | PMIC out is 3.3 Vdc |  |  |  |  |
| 3 | Connect the batter and the USB cable. | PMIC out is 3.3 Vdc |  |  |  |  |
| 4 | Connect a battery that is known to be 'dead' | Check the voltage of the battery every half hour until the battery's voltage has reached 4.2 Vdc. |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Writer:** Michael Walton | | | | | | |
| **Test Case Name:** | LTC4067 PMIC Test | **Test ID #:** | CMPS#1 | | | |
| **Description:** | The PMIC supplies the power to the system. The PMIC can have the battery, USB, or both as inputs. The outputs of the PMIC are a constant 3.3 Vdc and the charging of the battery. | **Type:** | White box | | | |
| **Tester information** | | | | | | |
|  | **Name of tester:** |  | **Date:** |  | | |
| **Hardware Version:** | 1.0 | **Time:** |  | | | |
| **Setup:** | The outputs of the PMIC must be tested with the three input combinations. The unit must output a constant 3.3 Vdc when connected to the battery, USB, or both. The PMIC must also charge the battery when its voltage has dropped below the 2.8 V lower threshold and stop charginge once the battery has a voltage of 4.2 Vdc. This test is to be conducted using a multimeter that is available in the labs on campus. | | | | | |
| **Step** | **Action** | **Expected result** | **Pass** | **Fail** | **N/A** | **Comments** |
| 1 | Connect the battery only. | The PMIC should output 3.3 Vdc |  |  |  |  |
| 2 | Disconnect the battery, plug in the USB cable attached to a DC source (computer) | PMIC out is 3.3 Vdc |  |  |  |  |
| 3 | Connect the batter and the USB cable. | PMIC out is 3.3 Vdc |  |  |  |  |
| 4 | Connect a battery that is known to be 'dead' | Check the voltage of the battery every half hour until the battery's voltage has reached 4.2 Vdc. |  |  |  |  |

# References